Kapitel 1 Bits

1.1 Signed og unsigned tal

Når en unsigned værdi konverteres til en signed værdi, er most significant bit, den bit der bestemmer om tallet er negativt eller positivt. Denne bit har værdien -8, når den er signed, og derved subtraheres der for alle yderligere bits.

Bits	Unsigned	Signed	Hex
0000	0	0	0x0
0001	1	1	0x1
0010	2	2	0x2
0011	3	2	0x3
0100	4	4	0x4
0101	5	5	0x5
0110	6	6	0x6
0111	7	7	0x7
1000	8	-8	0x8
1001	9	-7	0x9
1010	10	-6	0xA
1011	11	-5	0xB
1100	12	-4	0xC
1101	13	-3	0xD
1110	14	-2	0xE
1111	15	-1	0xF

Tabel 1.1: Tabel over signed og unsigned værdier

1.2 Operationer

For bits findes der operationer, som vist i DTG. Disse er bitvise, og noteres som vist i Tabel 1.2. For Aritmetiske skift, indsættes 1 på den nye plads, hvor ved Logiske skift, indsættes 0.

Operation	Navn	resultat
x|y	eller	1011 Da disse bit findes i enten eller af strengene.
x&y	og	0001 Da denne bit deles af de to strenge.
xŷ	XOR	1010 Da disse bits enten er i den ene eller den anden streng.
x	negation	1100 Da alle bits vendes.
y»3	Logisk højreskift	0001 Da bitværdien skifter 3 pladser mod højre.
x«3	Logisk venstreskift	1000 Da bitværdien skifter 3 pladser mod venstre.
y»2	Aritmetisk højreskift	1110 Da bitværdien skifter 3 pladser mod højre.
x«2	Aritmetisk venstreskift	1111 Da bitværdien skifter 3 pladser mod venstre.

Tabel 1.2: Bitvise operationer og deres resultater. Der er taget udgangspunkt i $x = 0011$, $y = 1001$

Kapitel 2 Assembly

Assembly sproget er det tætteste vi kommer på maskinkode. Derfor er det essentielt for computer arkitektur faget, at undervise i dette.

2.1 x64 Assembly

2.1.1 Registre

x64 Assembly kode har 16 64-bit registre at gøre godt med. Disse 16 registre kan ydermere opdeles i 32- 16- og 8-bit registre, som vist i Tabel 2.1

8-byte register	Bytes 0-3	Bytes 0-1	Byte 0
%rax	%eax	%ax	%al
%rcx	%ecx	%cx	%cl
%rdx	%edx	%dx	%dl
%rbx	%ebx	%bx	%bl
%rsi	%esi	%si	%sil
%rdi	%edi	%di	%dil
%rsp	%esp	%sp	%spl
%rbp	%ebp	%bp	%bpl
%r8	%r8d	%r8w	%r8b
%r9	%r9d	%r9w	%r9b
%r10	%r10d	%r10w	%r10b
%r11	%r11d	%r11w	%r11b
%r12	%r12d	%r12w	%r12b
%r13	%r13d	%r13w	%r13b
%r14	%r14d	%r14w	%r14b
%r15	%r15d	%r15w	%r15b

Tabel 2.1: Oversigt over registre i x64 Assembly

2.1.2 Ord og bytes

Indenfor Assembly arbejdes der med ord og bytes.

- En *byte* er 8 bits (10011011).
- Et *word* er 2 bytes (10011011 10010110).
- Et *dword* er 4 bytes og står for double word.
- Et *qword* er 8 bytes og står for quad word.
- Et *oword* er 16 bytes og står for octoword.

Bogstaverne *d* og *q* indgår også til tider i operationer, som eksempelvis *movq*. Dette betyder blot at der flyttes noget, der er 8 byte stort.

2.1.3 Instruktioner

Mange instruktioner som mov bruger suffix som *w* og *q* når disse bruges. Ydermere findes der flere forskellige former for operationer.

Data flytning

Instruktion	Source/Destination	Beskrivelse
`mov`	S, D	Flytter fra Source til Destination
`push`	S	Skubber til stakken
`pop`	D	Popper toppen af stakken til Destination
`mob`	S, D	Flytter byte til word (Sign)
`push`	S	Flytter byte til word (Zero)
`cwtl`		Konverterer word i `%ax` til dword i `%eax` (Sign)
`cltq`		Konverterer dword i `%eax` til qword i `%rax` (Sign)
`cqto`		Konverterer qword i `%rax` til oword i `%rdx:%rax`

Tabel 2.2: Instruktioner til at flytte data

Aritmetik

Instruktion	Source/Destination	Beskrivelse
`inc`	D	Inkrementerer Destination med 1
`dec`	D	Dekrementerer Destination med 1
`neg`	D	Aritmetisk negation
`not`	D	Bitvis komplement

Tabel 2.3: Unary operationer

Instruktioner som `leq 17(%rax, %rbx, 4), %rcx` skal forstås som `%rcx = 17 + %rax + %rbx *`

`leaq`	S, D	Indlæser effektiv addresse for Source til Destination
`add`	S, D	Adderer Source til Destination
`sub`	S, D	Subtraherer Source fra Destination
`imul`	S, D	Multiplerer Source med Destination
`xor`	S, D	Bitvis XOR Destination med Source
`or`	S, D	Bitvis OR Destination med Source
`and`	S, D	Bitvis AND Destination med Source

Tabel 2.4: Binære operationer

`sal/shl`	k, D	Venstreskift Destination med k bits
`sar`	k, D	Aritmetisk højreskift Destination med k bits
`shr`	k, D	Logisk højreskift Destination med k bits

Tabel 2.5: Binære operationer

`imulq`	S	Unsigned fuld multiplikation af `%rax` med S. Resultatet gemmes i `%rdx:%rax`
`mulq`	S	Signed fuld multiplikation af `%rax` med S. Resultatet gemmes i `%rdx:%rax`
`idivq`	S	Signed dividering af `%rdx:%rax` med S. Kvotient gemmes i `%rax`. Rest gemmes i `%rdx`
`divq`	S	Unsigned dividering af `%rdx:%rax` med S. Kvotient gemmes i `%rax`. Rest gemmes i `%rdx`

Tabel 2.6: Specielle Aritmetiske operationer

Kapitel 3 Øvelser

Kapitel 4

Kursusgang 9 Dette afsnit vil behandle Kursusgang 9 i CART

4.1 Practice Problems

I denne sektion vil de givne Practice Problems for Kursusgangen blive gennemgået.

4.2 Practice Problem 6.5

Reads		Writes	
Sequential read throughput	550 MB/s	Sequential write throughput	470 MB/s
Random read throughput (IOPS)	89,000 IOPS	Random write throughput (IOPS)	74,000 IOPS
Random read throughput (MB/s)	365 MB/s	Random write throughput (MB/s)	303 MB/s
Avg. sequential read access time	50 μs	Avg. sequential write access time	60 μs

Figur 4.1: Performance characteristics of a commercial solid state disk. Source: Intel SSD 730 product specifictations. IOPS is I/O per seconds. throughput numbers are based on reads and writes of 4 KB blocks. (Intel SSD 730 product specifictations. Intel Cooperation)

As we have seen, a potential drawback of SSDs is that the underlying flash memory can wear out. For example, for the SSD in Figur 4.1, Intel guarantees about 128 petabytes ($128 \cdot 10^{15}$) of writes before the drive wears out. Given this assumption, estimate the lifetime (in years) of this SSD for the following workloads:

1. *Wors case for sequential writes:* The SSD is written to continuously at a rate of 470 MB/s (The average sequential write throughput of the device).

2. *Worst case for random writes:* The SSD is written to continuously at a rate of 303 MB/s (the average random write throughput of the device)

3. *Average case:* The SSD is written to at a rate of 20 GB/day (the average daily write rate assumed by some computer manufacturers in their mobile computer workload simulations)

4.2.1 Udregninger til 6.5

Det er givet at $1PB = 10^9 MB$. Samtidig vides det at der er 86400 sekunder på en dag.

1. Med denne information kan følgende formel bruges til at udregne levetiden for en SSD ved worst case load:

$$\left(10^9 \cdot 128\right) \cdot \left(\frac{1}{470}\right) \cdot \left(\frac{1}{(86400 \cdot 365)}\right) = \frac{800000}{92637} \approx 8.6359 \tag{4.1}$$

2. Samme formel kan bruges til at udregne worst case for random writes

$$\left(10^9 \cdot 128\right) \cdot \left(\frac{1}{303}\right) \cdot \left(\frac{1}{(86400 \cdot 365)}\right) = \frac{8000000}{597213} \approx 13.396 \tag{4.2}$$

3. Formlen kan også bruges til at udregne vores average case. Dog skal sekunder ikke bruges i dette tilfælde, da der arbejdes med en tidsfaktor af dage.

$$\left(10^9 \cdot 128\right) \cdot \left(\frac{1}{20000}\right) \cdot \left(\frac{1}{365}\right) = \frac{1280000}{73} \approx 17534.24658 \tag{4.3}$$

Udregningerne er i år.